Common Biometric Exchange File Format

Revision History

1st Draft August 18, 1999

Technical Development Team

Paul Collier, Identicator
Jeff Dunn, NSA
Mark Jerde, Identicator
Larry O'Gorman, Veridicom
Fernando Podio, NIST/ITL
Lawrence Reineirt, NSA
Cathy Tilton, SAFLink

Note:
This document is a draft specification of a common biometric exchange file format. The specification is subject to change.

Table of Contents

TBD

Foreword

On February 21st 1999, the Information Technology Laboratory of the National Institute of Standards and Technology (NIST) and the US Biometric Consortium sponsored a Workshop to discuss the potential for reaching industry consensus in a common fingerprint template format. As a result of the discussions at the workshop, the participants agreed to develop a “technology-blind” biometric file format that will facilitate handling different biometric types, versions, and biometric data structures in a common way. The proposed effort does not include standardizing the content of these biometric data structures.

The following principles were generally agreed upon during this workshop:

- There is an advantage of a standard file format to facilitate exchange and inter-operability of biometric data.
- This format should include all modalities of biometrics. It should be "technology-blind". That is, it should not bias, encourage, or discourage any particular vendor or biometric technology from another.
- This format should not attempt to translate among different biometric technologies, but merely to identify them and facilitate their co-existence.

The target audience are developers of biometric-based systems and applications and users.

The expected benefits of the specification are: (a) reducing the need for additional software development that will identify different and proprietary biometric data structures supporting multiple biometric types within a system or application and (b) cost savings.

To ensure that the specification is in agreement with other biometric industrial efforts, the work is being coordinated with the BioAPI Consortium and other standardization activities undertaken by the industry, user organizations and the Biometric Consortium. Several BioAPI Consortium members are also participating in the development of this specification.

BioAPI Consortium Liaison:

Larry O’Gorman

Veridicom, Inc.

973-701-8700
log@veridicom

Next Workshop: September 17, 1999

On May 10th, 1999, NIST/ITL and the BC sponsored a Workshop to start development the common biometric exchange file format. The proposed effort does not include standardizing the content of these biometric data structures. During the May 10th Workshop it was agreed by the participants to continue the development of this file format in a second Workshop.

The second Workshop will be held on Friday, September 17, 1999, 9:00 a.m. to 5:00 p.m. at the Hilton Crystal City, Arlington, Virginia (703) 418-6800.

Participants need to sign a "Participant and/or Contributor's Agreement" in advance. If you need a copy of the agreement, please contact Fernando Podio (see below). Please fax the Workshop participant's agreement to Fernando Podio no later than September 1.

For any additional information about the Workshop or the biometric exchange file format, please contact:

Fernando Podio,
Information Technology Laboratory
NIST
Co-Chair Biometric Consortium
Bldg. 225/A248, Gaithersburg, MD 20899
(301) 975-2947
(301) 869-7429 (fax)
fernando.podio@nist.gov

Common Biometric Exchange File Format

1. Purpose

To establish an industry specification that defines a Common Biometric Exchange File Format and associated metadata that will enable interoperability of biometric-based application programs and systems from different vendors. This biometric file format will facilitate handling different types, versions, and technologies of biometric data in a common way. An application will easily recognize what type of biometric is available in the system, what version number, vendor's name, etc. and will be able to point at the proper biometric data. A common biometric file format may consist of a header that contains information such as file length and biometric types, followed by a block of data (data structure) in unspecified format that can pertain to one of any biometric template types and any other required biometric metadata.

The purpose of the specification is: (a) facilitating interoperability between different biometric technologies; (b) providings forward compatibility for technology improvements; and (c) simplifying the software/hardware integration process.

2. Scope

The specification intends to accommodate any biometric technology. The specification intends to include the definition of format and content for data elements such as: (a) a biometric metadata header that would contain such information as version number, length of data, encrypt, etc. for each biometric type available to the application/system; (b) biometric template data (content not specified); and (c) any other required biometric data or metadata structure. The biometric common format will not attempt to translate data among different biometric technologies, but merely will identify them and facilitate their co-existence. Although it is conceivable that industrial or user groups may agree upon common standard formats within the biometric data structures defined as part of this specification, the proposed effort does not include standardizing the content of these biometric data structures**.**

The current thinking is specifying one type of record:

It includes a Standard Biometric Header (SBH) and Standard Biometric Data Blocks (SBDBs). Each SBDB can include a block header identifying the type of data block and other required information such as the version number, length of data, and the data itself (e.g., data template).

This flexible and expandable file format allows for adapting the implementation to the biometric system or application the developer needs.

An application can use:

- A compliant small file with a minimum description (for small applications or systems that would not require more than a very brief description of the biometric data blocks) or

- A larger file including a larger number of descriptive fields and data structures as required by that application.

3. Conformance

TBD

4. References

TBD

5. Definitions

5.1 Global Unique Identifiers (GUIDs)

Global Unique Identifiers (GIUDS) will be used though out this document to identify a particular stucture or type of data (e.g. an identifier of a macthing algorithm). The GUID is a 128 bit value

Microsoft provides a GUID generator with most of its compiler tools (e.g. visual C++ or J++) called Guidgen.exe. It can be used to create a GUID.
An example of aGUID is as follows

// {FEBCD281-570A-11d3-BDBF-00C04F74F8A9}
static const GUID <<Sample.GUID>> =
{ 0xfebcd281, 0x570a, 0x11d3, { 0xbd, 0xbf, 0x0, 0xc0, 0x4f, 0x74, 0xf8, 0xa9 } };

These GUIDs are concidered to be unique values. Therefore no registration is required. Wherever a GUID is needed, the developer creates a GUID for that value and publishes it. Any application which integrates that product simply tries to match the GUID with its list of supported GUIDs products

6. Abbreviations and Acronyms

TBD

7. Data Structures

7.1 Introduction

The data structure defined in this document must be able to handle a wide variety of applications. Many current systems, which are using biometric verification, have limited storage media such as barcode, magstripe, and older smart cards. These applications require a data structure with minimum overhead.

Many newer systems are being developed which require a complex structure to incorporate enhanced features such as individual thresholding and reduction of false rejections.

The data structure specification is designed to accommodate these two extremes and intermediate solutions.

In order to achieve this requirement, the specified data structure shall have the following qualities:

- There must be a bare minimum structure which contains only the information necessary to process biometric data (ie. Minimize the overhead for those applications which don't need it).

- The Biometric Data type within the structure must be uniquely identified (to identify which algorithm(s) are need to perform the matching).

- Allow for current standards bodies to have flexibility to define data within the opaque data structure (e.g., The BioAPI Consortium).

- Allow for future expandability without complicated registration processes.

7.2 Proposed Data Structure

This data structure is one that has been used in many fields involving data exchange; a single, technology-neutral header followed by a technology-specific data block.

The format is as follows:

SBH - Standard Biometric Header

BSMB - Biometric Specific Memory Block

7.2.1 Standard Biometric Header (SBH)

The Standard Biometric Header is defined as shown in Table 1.

Table 1 - Standard Biometric Header

Field Name	Length (bytes)	Note
Record Type	3	3 character fixed field set to "BIO"
Header Version	1	Version of header, should be the version of this specification. Currently set to 00
Extension Type	1	00=Minim structure (No extension) 01=BioAPI Defined (See Appendix A) FF=GUID to follow
Extension GUID	16	Only present if SubType=FF
Data Format Type GUID	16	GUID of Data to follow
Data Size	4	Size of the Biometric Data to follow

Record Type: This Header is a 3 byte fixed field containing the phrase "BIO" encoded in ASCII.

Header Version: This 1 byte field allows for future revisions to this specification. Any implementation that wants to be compliant to this version of the specification will set this field to 00.

Extension Type: This 1 byte field allows for extensions to this header. Currently, the only extension defined for this revision of the specification is the BIOAPI extension (See Annex A for details). Any application can defined a compliant extension by setting this field to FF and populating the next field with a GUID of the extension. Future revisions of this document may define other extensions (02-FE).

Extension GUID: This field is used only if the Extension Type field is set to FF. It contains the GUID of the extension. It is outside the scope of this document to include any of these definitions. It is the Extension Developer's responsibility to document and maintain these extensions.

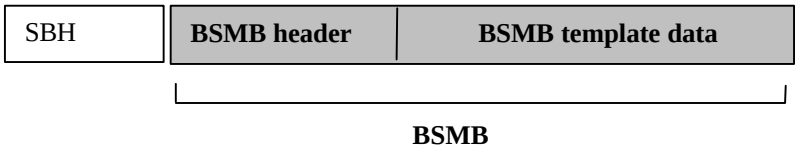
Data Format Type GUID: This GUID defines the GUID of the Biometric Data to follow. Annex B contains a sample GUID and definition for an X.509 Attribute certificate. It is outside the scope of this document to include any other definitions. It is the extension developer's responsibility to document and maintain these extensions.

Data Size: This 4 byte field holds the unsigned length of the data to follow this header.

7.2.2 Biometric Specific Memory Block (BSMB)

This is simply a block of memory that can be specified in any way by the owner of the type as specified in the *type* field of the SBH. Therefore, this can be a proprietary format or one agreed upon by a group.

The BSMB field format may not need any specification, there is likely to be a format analogous to the header/data format of most data storage structures. In this way, a vendor who "owns" this format can specify information in a header including version information, etc. Furthermore, it is conceivable, or likely, that groups may agree upon common standard formats within this BSMB level. In either case, BSMB's fields can be specified by a vendor or group.



- **BSMB header** - may contain such information as: version number, length of data, encrypt, etc.

- **BSMB template data** - block of memory containing template data.

7.3 The BioAPI Specification Version .1 BSMB header Extension

The BioAPI Steering Committee has released the data structure specified in the BioAPI Specification Version 0.1 as a contribution to the Common Biometric Exchange File Format

effort. BioAPI Consortium members can participate in the development of the Common Biometric Exchange File Format. It is anticipated that the BioAPI will create an extension compliant with this specification. Refer to Annex A of this specification for further detail.

7.4 Summary

A framework is shown here for a common biometric exchange file format. The file header defined in this document can be used by a wide variety of applications and can be expanded as needed within the conformance requirements of this specification.

ANNEX A
Extension Example

Type: BIO API Consortium

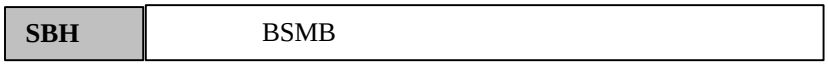
A.1 Introduction - Data structure proposed for the Common Biometric Exchange File Format by the BC Workshop

This data structure is one that has been used in many fields involving data exchange; a single, technology-neutral header followed by a technology-specific data block. (It has been included in this proposal with the understanding that this is only an example of how the file format to be developed under this effort might look like. It was included to encourage further discussions on the content of the required format.)

The format would look like,

SBH - Standard Biometric Header
BSMB - Biometric Specific Memory Block

Standard Biometric Header (SBH)



- **File Identifier Value ("magic number") -** first field of file identifies this as a biometric exchange file format (BEFF) type; a file identifier value is usually a long integer value or a short string of character bytes in length.

- **Length -** indicates length of data portion of file, BSMB, in bytes.

- **Type -** indicates BSMB type, e.g. "Company X", "Group Y", "Standard Z".

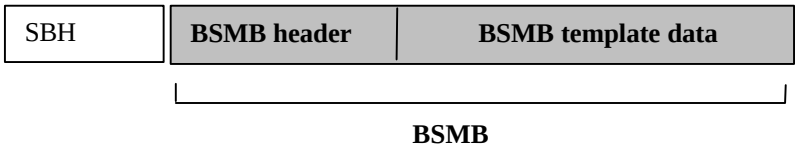
- **Time Stamp -** date and time (it is conceivable that a time stamp also appear in the BSMB data portion, which can be encrypted, so as to safeguard tampering of this information).

- **Encrypt Flag/Type -** indicates if the BSMB data portion is encrypted, and if so, by what method (this may or may not be needed in addition to the ability to encrypt within a portion of the BSMB, see below).

Biometric Specific Memory Block (BSMB)

This is simply a block of memory that can be specified in any way by the owner of the type as specified in the *type* field of the SBH. Therefore, this can be a proprietary format or one agreed to by a group.

BSMB field format may not need any specification, there is likely to be a format analogous to the header/data format of most data storage structures. In this way, a vendor who "owns" this format can specify information in a header including version information, etc. Furthermore, it is conceivable, or likely, that groups may agree upon common standard formats within this BSMB level. In either case, fields of BSMB can be specified by a vendor or group.



- **BSMB header** - may contain such information as: version number, length of data, encrypt, etc.

- **BSMB template data** - block of memory containing template data.

The example consist of a header that contains information such as file length and biometric type, followed by a block of data in unspecified format that can pertain to one of any biometric template type.

A.2 Data Structure Defined in the BioAPI Specification Version 0.1 (Type A)

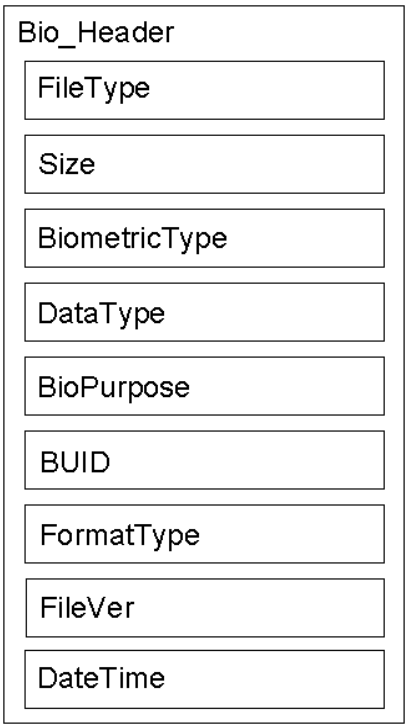
Note:
The BioAPI Steering Committee has released the data structure specified in BioAPI Specification Version 0.1 as a contribution to the Common Biometric Exchange File Format effort. BioAPI Consortium members can participate in the development of the Common Biometric Exchange File Format.

The data structures herein defined have been designed to be as flexible as possible, allowing the biometric vendor to store whatever information is needed, without unnecessary constraints. For example, the biometric data structures may contain a single biometric sample or may contain multiple samples. In order to support a wide range of process flow possibilities and biometric templates (models), these structures can be used to store any combination of data necessary to facilitate subsequent matching. It is the responsibility of the biometric technology provider (BSP) to fill this data structure with the data needed and in the format needed, and to be able to extract this data when it is needed.

A.2.1 Biometric Record Header

This data structure standardizes the header information preceding biometric data records to minimally and uniquely identify the content as well as to distinguish it from other, non-biometric data records.

Table A.1. Biometric record header



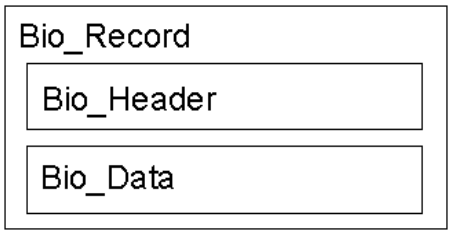
Member	Description
FileType	Unique, assigned value indicating that this as a Biometric Record. Constant string value = BIO.
Size	Unsigned long value indicating the size, in bytes, of the opaque biometric data block following the Bio_Header
BiometricType	Double word indicating biometric type (fingerprint, face, voice, iris, signature, hand, etc.)
DataType	Double word indicating data type as raw, intermediate, or processed.
BioPurpose	Double word indicating purpose of biometric data as identification or verification.
BUID	The BUID of the BSP which generated the data; stored in as a GUID structure (4 32-bit words).
FormatType	Double word indicating unique, specific format of biometric data; stored in OID format.
FormatRev	Revision level of FormatType specification (double word).
DateTime	The date and time at which the data structure was created or last updated. Format TBD.

A.3 Biometric Data

This data structure is vendor dependent; however, it is recommended that the vendor include header information within the data structure to facilitate its identification. BSPs are strongly

encouraged to document standard data types and formats within the raw biometric data or BIR structures, to facilitate and allow applications to make use of this data. Specifically, if raw biometric data is stored in BMP, TIFF, JPEG, GIF, WAV, AU or other standard formats, documentation should allow for the extraction and display of individual records.

Table A.2. Biometric data



Member	Description
Bio_Header	Standard biometric record header
Bio_Data	An opaque block of data containing raw or processed biometric data samples or templates (biometric identifier records – BIRs). The opaque data should begin with a vendor specific header further describing the contents. This subheader could be defined by another data format standard. Bio_Data can contain a combination of raw and processed data or may contain data captured from more than one biometric capture device.

Annex B
An EXAMPLE Data Format - The X.509 AuthenticationInfo Attribute certificate.

B.1 Background

This section has been appended as an example of a data format type. It is widely believed that many systems will, in the future, use X.509 certificates to hold biometric templates, therefore this may be an appropriate example. This data can be used in any Extension type (minimal, BioAi , or externally defined).

B.2 The X.509 Certificate

The X.509 standard is an international standard for security and authentication services supporting security frameworks for electronic information distribution. The term "X.509 certificate" has become the defacto name for public key certificates in use today. This specification defines the main data structure (i.e. the "certificate") used for performing these services and addressing the handling of keys.

The X.509 certificate is primarily used to hold public key information. In addition to public key information, X.509 also describes certificate revocations lists (CRLs) which are signed lists of certificate serial numbers which have been compromised. A large portion of the system which utilize public key certificates may be focused on the managing of the certificates via CRLs. The creation, distribution, and utilization of CRLs can become quite complex. Systems which utilize multiple certificates can greatly increase the complexity of the system if different CRLs are used to control each type of certificate. When designing the architecture of a system, the management burden to support the system must be considered. The focus of this document, however, will be on how to utilize certificates to provide authentication techniques, not CRL management.

X.509 is a recommendation by the International Telecommunications Union (ITU). X.509 is equivalent to the International Standards Organization (ISO) / International Electrotechnical Commission (IEC) 9594-8 specification. The X.509 specification and ISO/IEO 9594-8 document are identical. X9.55, originated by the American Bankers Association (ABA) and adopted by ANSI, contains an equivalent description from the perspective of the banking industry. ISO/WD-15782 is the international version of the banking industry's certificate management descriptions, which borrows from X9.57 and X9.55.

Information contained in the X.509 Version 3 Certificate is as follows:

Table B.1 - X.509 Version 3 Fields

Field	Description
version	This identifies which version of the X.509 standard applies to this certificate. This affects what information can be specified in it. Version 1 has been available since 1988, and is widely deployed. (Note: a value of 0 indicates Version 1, 1 indicates Version 2, and 2 indicates Version 3)
serial Number	The issuer creates a unique serial number and places it here. It is intended to distinguish this certificate from all others created by the issuer.
subject	The name of the entity (usually a human) whose public key the certificate identifies. This name uses the X.500 standard's Distinguished Name (DN), and is intended to be unique. The DN is comprised of such information as the country, region, and given name of the entity.
subject Unique Identifier	The unique id of the subject (optional for version 3 and after). Added in version 2, this field provides a location to specify a bit string to uniquely identify the entity's X.500 DN, in the event that the same DN has been assigned to more than one entity over time.
subject PublicKeyInfo	The Public Key field consists of: The ID of the algorithm the public key is to be used with, and the public component of the key which belongs to the entity specified by the Subject Name. This value is used to encrypt information to be sent to the entity.
issuer	The name of the entity that is signing the certificate. The entity is usually a Certificate Authority (CA). This name uses the X.500 standard (the Distinguished Name), and is intended to be unique. The CA can sign its own certificate.
issuer Unique Identifier	The unique id of the issuer (optional). Added in version2, this field provides a location to specify a bit string to uniquely identify the issuer X.500 DN, in the event that the same DN has been assigned to more than one CA over time.
validity	This specifies when a certificate is valid. This period is described by a start date and time and an end date and time as follows: NotBefore: The start time that the certificate is valid. notAfter: The end time that the certificate is valid.
extension(s)	Add on fields (Optional-see following section)
signature	The signature field consist of: Identifier of the algorithm used to create the signature and the output of the signing function (i.e. the signed hash value of the data in this certificate). This data is used to verify the data in the certificate.

B.3 Attributes

The following is a discussion on attributes that can be used in X.509 certificates. Attributes are defined in X.501 as "information of a particular type". The attribute describes a characteristic of an associated object. Refer to X.501, reference [17] for further information about attributes.

The AttributeTypeAndValue referenced by the RelativeDistinguishedName (RDN) is defined as follows:

```
AttributeTypeandValue::=SEQUENCE
   type   ATTRIBUTE.&id ({SupportedAttributes});
   value ({ATTRIBUTE.&Type({SupportedAttributes}){@type})}
```

Where the ATTRIBUTE.&id is the object identifier assigned to it and the ATTRIBUTE.&TYPE is the it attribute syntax (An ASN.1 type such as BIT STRING, INTEGER, etc.). Both the id and Type are defined in a class named ATTRIBUTE.

The Attribute construct utilized by X.509 related constructs is defined as follows:

```
Attribute  ::=          SEQUENCE {
           type         ATTRIBUTE.&id ({ SupportedAttributes}),
           values       SET SIZE (0.. MAX) OF ATTRIBUTE.&TYPE ({ SupportedAt-
tributes}{@type}),
           valuesWithContextSET SIZE (1 .. MAX) OF SEQUENCE {
                        value    ATTRIBUTE.&Type ({SupportedAttributes}{@type})
OPTIONAL,
                        contextList       SET SIZE (1 .. MAX) OF Context} OPTIONAL}
```

Contexts are properties of the attribute which are used to determine the applicability of the attribute. As an example, contexts can be used to associate a particular language, time, or locale.

The Context is defined by X.501 as follows:

```
Context::= SEQUENCE {
        contextType             CONTEXT.&id ({SupportedContexts}),
        contextValues  SET SIZE (1..MAX) OF CONTEXT.&Type ({SupportedCon-
texts}{@contextType}),
        fallback          BOOLEAN DEFAULT FALSE}
```

All attributes must be standardized (i.e. registered) with an ISO recognized standards body. Some are registered with international organizations, such as the ISO, and other are registered at the national level organizations, such as ANSI. There is no single source or document that lists all registered attributes.

Some types are listed in X.520 along with a description of the value (BIT STRING, INTEGER, etc.). Even with the description given in X.520, there is still room for interpretation of the fields. There is not always a real standard as to the semantics and exact meaning of each of the elements in the RDNSequence. As an example, some name creating bodies will use the CommonName and others will use a combination of surName and givenName.

X.521 (equivalent to ISO 9594-7) defines several groupings of attributes or "classes". The X.509 certificate can make use of these classes.

Refer to ITU X.509 or ISO/IEO 9594-8 for further details on the X.509 Certificate definitions or X.501 and X.520 for further definitions of the Distinguished Name.

B.4 Attribute Certificates

B.4.1 Introduction

Attribute certificates are used to convey a set of attributes along with a public key certificate identifier (i.e. a serial number and a public key certificate issuer name) or entity name. The attributes are placed in a separate structure to maintain conformance with existing international standards (X.509). An entity may have multiple attribute certificates associated with each of its public keys certificates.

X9.57, originated by the American Bankers Association (ABA) and adopted by ANSI, also defines an attribute certificate which is complimentary to the X.509 certificate. ISO/IWD-15782-2 [2] also describes the attribute certificate, with added details for banking applications.

There is no requirement that the same authority create both the public key certificate and the attribute certificate; in fact, role separation should frequently dictates otherwise. The generation of an attribute certificate may be requested by an entity other than the subject of the attribute certificate. The X9.57 specification does not define the messages between an entity and the attribute authority (AA) dealing with the generation of the attribute certificate.

X9.57 defines an attribute as information, excluding the public key, which is provided by an entity or an AA and certified by the AA in an attribute certificate. Attributes are bound to a public key certificate or entity name by the signature of the AA on the attribute certificate.

The information contained in the attribute certificate is as follows:

Table B.2 - X9.57 Attribute Certificate Fields

Field	Description
Version	This identifies the version of the attribute certificate.
serial Number	This field uniquely identifies this certificate among all those issued by the AA. (if the AA is also a CA, the serial number space is thus shared by the public key certificates and the attribute certificates.)
owner	An attribute certificate may be linked to either a particular entity, or one of that entity's public key certificates. The mechanism to be used is specified by the application or standard which uses the attribute certificate.
IssuerName	This field contains the name of the issuer of the attribute certificate (an AA).
Issuer Unique Identifier	This field uniquely identifies the issuer, in the case where the issuer name is not sufficient.
validity	This specifies when a certificate is valid. The period is described by a start date and time and an end date and time as follows: notBefore: The start time that the certificate is valid. notAfter: The end time that the certificate is valid.
Attributes	The attributes are information concerning the entity, or the certification process. They may be supplied by either the entity, a third party entity or the AA depending upon the application.
extension(s)	The extensions field allows addition of new fields to the attribute certificate without modification of the ASN.1 definition.
signatureAlgorithm	This field identifies the algorithm used to sign the certificate.
signature	The signature field consists of: The output of the signing function (i.e. the signed hash value of the data in this certificate). This data is used to verify the data in the certificate.

The AttributeCertificate matching rule was created to allow more complex matching than the certificateExactMatch (a matching rule defined in X.509). It allows comparison to the issuer's serialNumber, the owner, the issuerName, and the validity. Refer to X.509 for further information on the matching rules.

A new attribute type of AttributeCertificate is defined to bind the attribute certificate to an X.509 certificate or directly to a subject name. The AttributeCertificateAttribute can be used with the certificateExactMatch to verify the binding.

B.4.2 Attribute Certificate Advantages/Disadvantages

Attribute certificates are essentially X.509 certificates without public key information (alternatively one can perceive them as extended certificates without the X.509 certificate embedded into them.) They are intended to compliment the X.509 certificate with additional information about the user (subject). This would give the same advantages and disadvantages as the PKCS#6 certificate with the additional benefits and disadvantages listed below:

Advantages:

- Mutual verification, via a challenge response, can be performed between the holder of the attribute certificate and the user authenticator prior to sending the attribute information.
- The attribute information can be encrypted, providing access to the confidential information to verified authenticators only.
- Information can be separated into as many attribute certificates as needed by the system. This may be useful in meeting the "need to know" requirement of many systems.
- Anonymity can be accommodated if the Distinguished Name (DN) of the user's X.509 certificate is a reference, not an actual identity (i.e. a user number, database lookup, etc.). The DN can be used to match attribute certificates with X.509 certificates.
- Attribute certificates are becoming standardized (as with X.509).

Disadvantages:

- Introducing multiple attribute authorities into the system architecture makes the system more complex. Key management issues may be prevalent.
- User authentication processing time may be an issue if two signatures must be verified, and the attribute certificate needs to be decrypted.

B.5 Data Format - The X.509 AuthenticationInfo Attribute certificate

B.5.1 Creating the Header

The Header is created as specified in section 1.3. For this example we will create a minimal Extention type and a Certificate which size is 0x346 (838 decimal) . The steps consist of assigning a GUID and creating the Header structure.

B.5.1.1 Assigning the GUID

The GUID created by Microsoft™ supplied GUID generator (guidgen.exe) is as follows:

```
// {FEBCD281-570A-11d3-BDBF-00C04F74F8A9}
static const GUID <<Sample.GUID>> =
{ 0xfebcd281, 0x570a, 0x11d3, { 0xbd, 0xbf, 0x0, 0xc0, 0x4f, 0x74, 0xf8, 0xa9 } };
```

Therefore the GUID will contain the value "**febcd281570a11d3bdbf00004f74f8a9".** This is the value to place in the Format Type field of the header.

B.5.1.2 Creating the Header structure

The header structure will not contain an extension type GUID since this example is using a mininal extension type. The header values are as follows:

Table B.3 – Header Structure

Field Name	Length	Value
Record Type	**3**	**"BIO"**
Header Version	**1**	**00**
Extention Type	**1**	**00**
Data Format Type GUID	**16**	**febcd281570a11d3bdbf00004f74f8a9**
Data Size	**4**	**346**

The data that follows this header is desribed in the next subsection.

B.5.1.3 Creating the Certificate

Encoding Rules

The smartcard will be required to verify the signature on certificates loaded into it for authentication purposes. Since smartcards are limited in storage, processing power, and other resources, it is suggested that Packed Encoding Rules (PER) [Cert1] be used for all certificates from the certificate authorities and user certificates loaded into the smartcard.

Signature Certificate Processing

The smartcard will be required to store all CA certificates required for authentication. These certificates will be used to verify certificates loaded into the smartcard [Cert2]. This includes any CA certificates, user signature and/or key exchange keys, and authentication certificates. This implies that the smartcard must be capable of decoding each certificate loaded into itself and verify signatures [Cert3].

Using Attribute Certificates

ECMA.219 described the methods used to provide the authentication functions. The Subject Directory Attribute Extension, the attribute certificate, and the PKCS Extended Certificate[6] all make use of (registered) attributes. The ECMA defined methods and currently registered attributes could be used without modification and be placed in an X.509 certificate or be placed directly in the locations described reference [TBD] as long they are the only authentication techniques that the system employing the authentication information needs. If the system utilizes AI such as a biometric, then more information is needed.

B.6 X.509 Attributes

X.509 imports the attribute definition from X.501.

The X.501 defined attribute (that is AttributeTypeandValue) is as follows:

```
AttributeTypeandValue::=SEQUENCE
   type   Attribute.&id ({SupportedAttributes});
   value ({Attribute.&Type({SupportedAttributes}){@type})}
```

All attributes are assigned an identifier using an object type of id-at. Any registered attribute, assigned a unique identifier by an ISO recognized standards body, can be used. X.520 is a source for ISO defined attributes; however, many other standards body have registered attributes which may used.

There as several authentication information attributes already defined. Among them are:

" userPassword OBJECT IDENTIFIER::= {id-at 35}

" userCertificate OBJECT IDENTIFIER::= {id-at 36}

" x121Address OBJECT IDENTIFIER::= {id-at23}

These attributes can be placed directly in the locations described reference [6] as long they are the only authentication techniques the system needs.

B.6.1 Defining a new Attribute

To utilize the authentication information in an X.509 or related certificate, the authentication information would have to be defined as an attribute. If the authentication information construct, as defined in ECMA.219, is given the attribute syntax, the following attribute is the result:

```
authenticationInfo ATTRIBUTE::={
        WITH SYNTAX         AuthenticationInfo,
        ID                  id-at-TBD}
AuthenticationInfo::=SEQUENCE{
        authenticationMethod[0] AuthenticationMethod, -- defined in ECMA.219
        exchangeAI[1]AuthMparm, -- defined in ECMA.219
        biometricInfo BiometricInfo OPTIONAL -- defined in section 2.3.2.2 of this document}
```

Since all the attribute placement options discussed in [6] can have multiple attributes per use, it is suggested that only one information object be placed in an attribute, if multiple objects are used. This implies if the application is using five fingerprints per user, that each fingerprint be placed in a separate AuthenticationInfo attribute. This will allow the application using the information to scan the attributes and select the appropriate one.

B.6.2 Additional Authentication Methods

As technology advances, additional authentication methods will be developed. If the methods are not listed in ECMA.219, then the authentication method should be register with the appropriate registry authority. Such additional methods may include:

Handwriting Recognition: An individual is prompted to write random phrases on a touchpad. This data is then compared to samples in a database.

Hand Geometry: An immutable characteristic which utilizes the bone structure of the hand to verify individuals.

Facial Recognition: An immutable characteristic which utilizes facial characteristics to verify individuals.

B.6.3 Additional Detailed Biometric Information

B.6.3.1 Biometric information fields

There are many different processes capable of performing biometric verification under a specific method (i.e. there can be different processes for performing fingerprint verification). In order to allow for the continual evolution of biometric products, a data structure must be defined which can hold varying formats of data and information needed to identify the information for correct processing. This data structure must be capable of evolving with the biometric technology and it must be capable of customizing by the applications which may utilize a subset of the information.

Such a structure may contain the fields shown in Table B.3.

Table B.4 - Biometric information fields

Field Name	Description
ProcessingInfo	These (optional) fields are used to provide information to the process(es) which are used to create a livescan biometric template to compare against the biometric template found in this structure.
MatchingAlgorithmInfo	These fields specify which algorithm(s), and their respective versions, the biometric template is compatible with. They also provide any algorithm specific information.

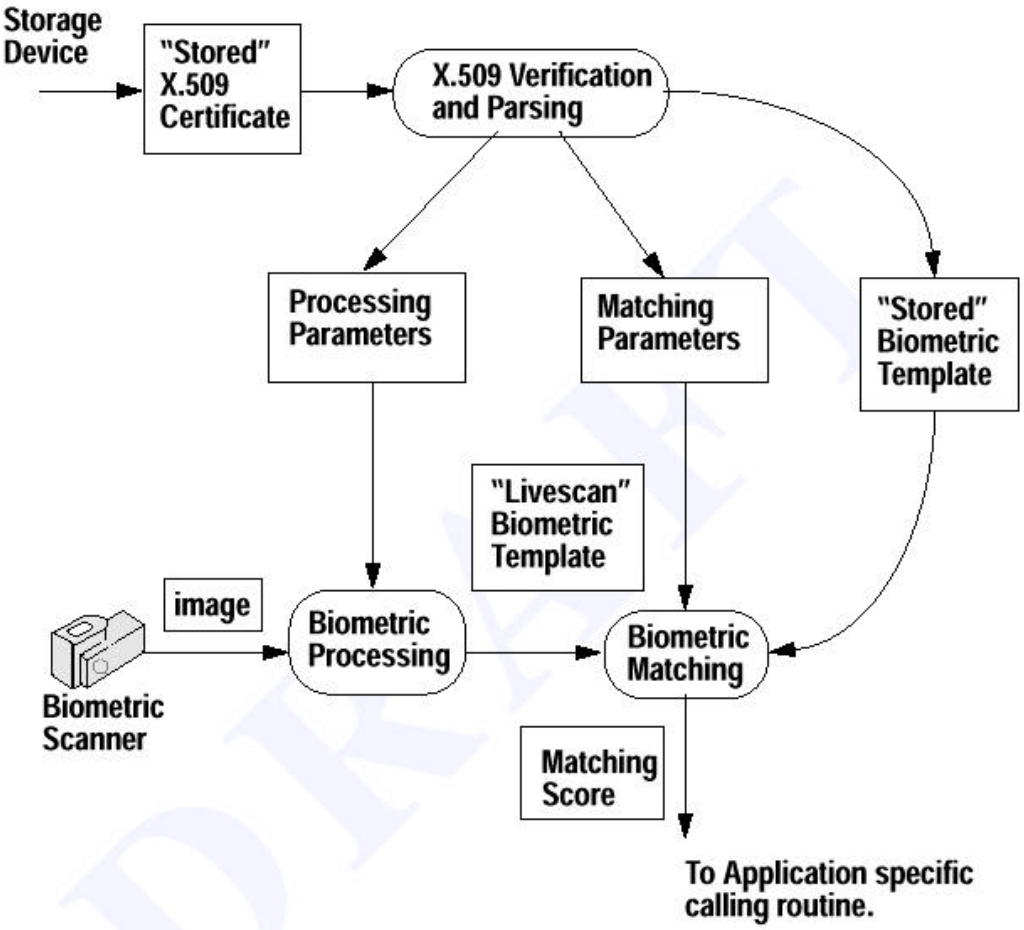
Where each of the information fields will consist of the following data:

Table B.5 – Data contained in the biometric information fields

Field Name	Description
ID	Each biometric process or matching algorithm must be registered by the manufacturer with the appropriate standards body. This will insure a unique identifier for application to determine compatibility.
Parameters	Each manufacture that registers its process or matching algorithm must document the parameters which are used with it and can be placed in a certificate.
Version	The version of the component used to create the information. Note: the decision of compatibility with the information will be left to the application.

The following diagram illustrates how the biometric processing and matching parameters would be utilized during a biometric verification process:

Figure B.1: Using an X.509 Certificate with Detailed Biometric Information



The certificate used to store the biometric information would be transferred to the entity performing the verification (from a database, smartcard, disk, etc.). The entity would verify the signature on the X.509 certificate to detect alteration and to prove the validity of the biometric template. The processing parameters are fed to the biometric processing function which converts the livescan image to a livescan biometric template. The livescan biometric template, the biometric template from the X.509 certificate and the matching algorithm parameters from the X.509 certificate are fed into the matching algorithm for verification of the user.

The ASN.1 Syntax of the Biometric information would be as follows:

```
BiometricInfo::=SEQUENCE{
processingInfo          ProcessingAlgorithmInfo OPTIONAL,
matchingInfo            MatchingAlgorithmInfo}
```

Where the ProcessingInfo and MatchingInfo are defined in the following sections.

B.6.3.2 Processing Information

Biometric processing is the function which takes a biometric sample (typically a video image or an audio sample), extracts information from the sample (such as a location of the minutia in the fingerprint), and creates a output file (typically called a biometric template). The processing information field would be used to provide processing algorithm specific information which may be used to personalize the process for the individual. The algorithm used to create the biometric template is specified by the processingAlgorithmID. processAlgorithmParams is used to provide process specific parameters.

The syntax for this field would be defined as:

```
ProcessingInfo::=SEQUENCE{
processingID            OBJECT IDENTIFIER, -- Defined by implementation
processingParms         AuthMparm OPTIONAL, -- Defined in ECMA.219
processingVersion       Version} -- Defined in X.509
```

Version is defined in X.509 as follows:

```
Version::=INTEGER{v1(0)}        -- Add versions as needed –
```

B.6.3.2.1 Registering Biometric Processes

The need to identify this process is negligible from a certificate's point of view, unless the process creating the livescan sample to compare against the certificate requires some customizing in regard to the individual who is being sampled. If this is the case, then the individual process creating the template needs to be registered by a recognized standards body. The OBJECT IDENTIFIER assigned during the registration process will be used to associate the parameters defined below.

B.6.3.2.2 Biometric Processing Parameters

As stated above, biometric processes only needs to be registered if there are associated parameters that need to be sent. The individual processing parameters do not have to be registered as long as they are defined and maintained by the organization which registered the process. The processing parameters are associated with that particular OBJECT IDENTIFIER.

Examples of processing parameters may be:

Minimal Acceptable Quality: A minimum quality that the sample must have to be accepted for further processing (useful if the particular biometric can obtain preliminary quality ratings on a sample). This may relieve the need for users with poor biometric characteristics (such as a scarred finger) to reenter a biometric sample several times for verification.

Number of Samples: The number of samples that should be taken of the user which meet the MinimumAcceptableQuality threshold. This will also help users with poor biometric characteristics to avoid reentering a biometric sample several times.

The processing parameters could follow the established by ECMA.219 AuthMparm or define an entirely new syntax using the ANY option.

The application would be responsible for determining compatible versions. If the versions are incompatible, then the processing information may have to be rejected, and therefore the authentication process would have to fail.

Such parameters should and could be standardized upon to reduce the overhead for systems which want to incorporate multiple biometric devices. This is likely to happen in the future as biometric technology matures.

Biometric matching is the function (algorithm) which takes two biometric templates and compares them for similarities. The output of the matching function is typically a matching score representing the amount of similarity found between the two templates.

The biometric templates are generally designed to work with a specific biometric matching algorithm. The application can reference the ID of the MatchingInfo in this field to determine compatibility.

Parameters for the matching algorithm may be supplied to provide matching algorithm specific information to personalize the matching process for the individual. The algorithm used to match the biometric template is specified by the matchingAlgorithmID. processAlgorithmParams is used to provide process specific parameters (such as matching thresholds).

The syntax for this field is defined as follows:

```
MatchingInfo::=SEQUENCE{
        matchingID              OBJECT IDENTIFIER, -- Defined by implementation
        matchingParm            AuthMparm OPTIONAL, -- Defined in ECMA.219
        matchingVersion         Version} -- Defined in X.509
```

Version is defined in X.509 as follows:

```
Version::=INTEGER{v1(0)}        -- Add versions as needed –
```

B.6.3.2.3 Registering Biometric Matching Methods

The currently available biometric related Software Development Kits (SDK) offer company/ device specific routines and data. Although there may be standardized Application Programming Interfaces (APIs) developed in the future, there are enough existing proprietary devices and methods in circulation to justify the need to differentiate between the various implementations. If a system utilizes more that one biometric device in the system, there must be a way to choose which software to use. Registering the biometric matching method would be a means of doing so.

The individual process which matches the biometric template placed in the X.509 certificate against the livescan biometric template needs to be registered by a recognized standards body.

The OBJECT IDENTIFIER assigned during the registration process will be used to associate the parameters defined below.

B.6.3.2.4 Biometric Matching Parameters

As mentioned above, the matching identifier can be used to determine compatibility between the X.509 certificate and the system being accessed. If there are any parameters required for the matching process then the matching parameters can be used to provide this information securely.

An example of a matching parameter may be:

> Minimal Acceptable Matching Threshold: This may be used to adjust the sensitivity of the matching process for individual with poor biometric characteristics.
>
> Template Identifier: This may be used to distinguish the biometric template from other found in certificate. It may be used to for such distinctions as which finger the template represents (index finger, thumb, etc.).

The matching parameters would be defined and maintained by the organization which registered the matchingID. The application would be responsible for determining which versions it has compatibility with. If the versions are incompatible, then the matching information may have to be rejected, and therefore the authentication process would have to fail.

As with the biometric processing parameters, these parameters should and could be standardized upon to reduce the overhead for systems which want to incorporate multiple biometric devices.

B.7 ASN.1 Authentication Information (AI) Attribute Definition

B.7.1 Atribute syntax

There currently exists no current attribute definition that can be used in a X.509 or related certificate if a system requires multiple types of authentication information. By using some of the Authentication Information definitions provided by ECMA.219, we can construct a new attribute for placing the information in an X.509 certificate. The complete syntax of the attribute would be as follows:

```
authenticationInfo ATTRIBUTE::={
        WITH SYNTAX         AuthenticationInfo,
        ID                  id-at-TBD}

AuthenticationInfo::=SEQUENCE OF{
        authenticationMethod [0] AuthenticationMethod, -- defined in ECMA.219
        exchangeAI     [1]AuthMparm, -- the data, as defined in ECMA.219
        biometricInfo BiometricInfo OPTIONAL -- defined in section 2.3.2.2 of this document}

BiometricInfo::=SEQUENCE OF{
         processingInfo                 ProcessingInfo OPTIONAL,
         matchingInfo           MatchingInfo}
```

```
ProcessingInfo::=SEQUENCE OF{
        processingID        OBJECT IDENTIFIER, -- Registered by implementation
        processingParms             AuthMparm OPTIONAL, -- Defined in ECMA.219
        processingVersion           Version}        -- Defined inX.509

MatchingInfo::=SEQUENCE OF{
        matchingID          OBJECT IDENTIFIER, -- Registered by implementation
        matchingParm                AuthMparm OPTIONAL, -- Defined in ECMA.219
        matchingVersion     Version}   -- Defined inX.509

--- ECMA.219 definitions

AuthMparm::=CHOICE{
        printableValue      [0] Printable String,
        integerValue   [1] INTEGER,
        octetValue     [2] OCTET STRING,
        bitStringValue [3] BIT STRING,
        otherValue     [4] ANY} -- defined by authenticationMethod (i.e. the AI)

-- X.509 Definitions
Version::=INTEGER{v1(0)}          -- Add versions as needed --
```

Each implementation of a biometric matching algorithm and biometric processing functions would require registration with an appropriate standards body. The processing functions would only have to register identifiers if they need to send parameters to the processing function which takes a livescan for comparison.

For standardization, the AuthenticationInfo attribute must be registered along with the provided, or a similar, definition. The registration of this attribute is beyond the scope of this document.

B.7.2 ASN.1 Authentication Attribute Certificate definition

The attribute certificate that holds the authentication information attribute is described in ASN.1 as follows [Cert 4]:

```
AttributeCertificate ::= SIGNED {AttributeCertificateInfo}

AttributeCertificateInfo ::= SEQUENCE {
            version Version DEFAULT v1,
            subject CHOICE {
            baseCertificateID[0]  IssuerSerial, -- associated with a Public Key Certificate
                    subjectName[1]GeneralNames },  -- associated with a name
            issuer                 GeneralNames, -- CA issuing the attribute certificate
            signature              AlgorithmIdentifier,
            serialNumber          CertificateSerialNumber,
```

```
        attrCertValidityPeriod  AttCertValidityPeriod,

        authenticationInfo      AuthenticationInfo,

        UniqueissuerID,

        issuerUniqueID          UniqueIdentifier OPTIONAL,
        extensions              Extensions OPTIONAL}

    IssuerSerial  ::= SEQUENCE {
        issuer          GeneralNames,
        serial                  CertificateSerialNumber,
        issuerUID               UniqueIdentifier OPTIONAL}

AttCertValidityPeriod ::= SEQUENCE {

        notBeforeTime GeneralizedTime,
        notAfterTime   Generalized Time }
```

B.8 Approximate Certificate Data Size

An approximation of data sizes can be made on the following assumptions.

" The size of the signature and public key info is set at 512 bits (64 octets where 1octet = 1 byte).

" No extensions are used.

" Packed Encoding Rules (PER) is utilized by the CA signature certificates and user certificate.

Table B.6 - Contents of Attribute Cert - Identification & Authentication Cert

Item	item size	# of items	Total Size
Version	**5 octets**	**1**	**5 octets**
Owner (baseCertificateID)	**8 octets**	**1**	**8 octets**
issuer (AA)	**183 octets**	**1**	**183 octets**
signature	**9 octets**	**1**	**9 octets**
serialNumber	**6 octets**	**1**	**6 octets**
validity	**32 octets**	**1**	**32 octets**
AuthenticationInfo -biometricInfo	**500 octets**	**1**	**500 octets**
IssuerUniqueID (Token Serial #)	**16 octets**	**1**	**16 octets**
AlgorithmIdentifier	**9 octets**	**1**	**9 octets**
signatureValue	**70 octets**	**1**	**70 octets**
Total			**838 octets**

If additional fields are added (such as extensions) simply add the length of the new field to the total.

Table B.7 - Contents of $Cert_{Host}$

Item	Item size	# of items	Total Size
Version	**5 octets**	**1**	**5 octets**
SerialNumber	**6 octets**	**1**	**6 octets**
Signature	**9 octets**	**1**	**9 octets**
Issuer	**183 octets**	**1**	**183 octets**
Validity	**32 octets**	**1**	**32 octets**
Subject	**183 octets**	**1**	**183 octets**
SubjectPublicKeyInfo	**82 octets**	**1**	**82 octets**
AlgorithmIdentifier	**9 octets**	**1**	**9 octets**
SignatureValue	**70 octets**	**1**	**70 octets**
Total			**579 octets**